CORE
Metadata, citation and similar papers at core.ac.uk
Provided by The International Islamic University Malaysia Repository

Chapter 15

Grammars Controlled by Petri Nets

J. Dassow, G. Mavlankulov, M. Othman, S. Turaev, M.H. Selamat and R. Stiebe

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/50637

1. Introduction

Formal language theory, introduced by Noam Chomsky in the 1950s as a tool for a description of natural languages [8–10], has also been widely involved in modeling and investigating phenomena appearing in computer science, artificial intelligence and other related fields because the symbolic representation of a modeled system in the form of strings makes its processes by information processing tools very easy: coding theory, cryptography, computation theory, computational linguistics, natural computing, and many other fields directly use sets of strings for the description and analysis of modeled systems. In formal language theory a model for a phenomenon is usually constructed by representing it as a set of words, i.e., a *language* over a certain alphabet, and defining a generative mechanism, i.e., a *grammar* which identifies exactly the words of this set. With respect to the forms of their rules, grammars and their languages are divided into four classes of *Chomsky hierarchy*: *recursively enumerable*, *context-sensitive*, *context-free* and *regular*.

Context-free grammars are the most investigated type of Chomsky hierarchy which, in addition, have good mathematical properties and are extensively used in many applications of formal languages. However, they cannot cover all aspects which occur in modeling of phenomena. On the other hand, context-sensitive grammars, the next level in Chomsky hierarchy, are too powerful to be used in applications of formal languages, and have bad features, for instance, for context-sensitive grammars, the emptiness problem is undecidable and the existing algorithms for the membership problem, thus for the parsing, have exponential complexities. Moreover, such concepts as a derivation tree, which is an important tool for the analysis of context-free languages, cannot be transformed to context-sensitive grammars. Therefore, it is of interest to consider "intermediate" grammars which are more powerful than context-free grammars and have similar properties. One type of such grammars, called *grammars with regulated rewriting* (*controlled* or *regulated grammars* for short), is defined by considering grammars with some additional mechanisms which extract some subset of the generated language in order to cover some aspects of modeled phenomena. Due to the variety of investigated practical and theoretical problems, different additional mechanisms to grammars can be considered. Since Abraham [1] first defined matrix grammars in 1965, several grammars with restrictions such as programmed, random

INTECH
open science | open minds

©2012 Turaev et al., licensee InTech. This is an open access chapter distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0),which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

context, valence grammars, and etc., have been introduced (see [16]). However, the rapid developments in present day technology, industry, medicine and other areas challenge to deal with more and more new and complex problems, and to look for new suitable tools for the modeling and investigation of these problems. *Petri net controlled grammars*, which introduce *concurrently parallel control mechanisms* in formal language theory, were proposed as a theoretical model for some problems appearing in systems biology and automated manufacturing systems (see [18–23, 56, 59–62]).

Petri nets, which are graphical and mathematical modeling tools applicable to many concurrent, asynchronous, distributed, parallel, nondeterministic and stochastic systems, have widely been used in the study of formal languages. One of the fundamental approaches in this area is to consider Petri nets as language generators. If the transitions in a Petri net are labeled with a set of (not necessary distinct) symbols, a sequence of transition firing generates a string of symbols. The set of strings generated by all possible firing sequences defines a language called a Petri net language, which can be used to model the flow of information and control of actions in a system. With different kinds of labeling functions and different kinds of final marking sets, various classes of Petri net languages were introduced and investigated by Hack [34] and Peterson [46]. The relationship between Petri net languages and formal languages were thoroughly investigated by Peterson in [47]. It was shown that all regular languages are Petri net languages and the family of Petri net languages are strictly included in the family of context-sensitive languages but some Petri net languages are not context-free and some context-free languages are not Petri net languages. It was also shown that the complement of a free Petri net language is context-free [12].

Another approach to the investigation of formal languages was considered by Crespi-Reghizzi and Mandrioli [11]. They noticed the similarity between the firing of a transition and application of a production rule in a derivation in which places are nonterminals and tokens are separate instances of the nonterminals. The major difference of this approach is the lack of ordering information in the Petri net contained in the sentential form of the derivation. To accommodate it, they defined the commutative grammars, which are isomorphic to Petri nets. In addition, they considered the relationship of Petri nets to matrix, scattered-context, nonterminal-bounded, derivation-bounded, equal-matrix and Szilard languages in [13].

The approach proposed by Crespi-Reghizzi and Mandrioli was used in the following works. By extending the type of Petri nets introduced in [11] with the places for the terminal symbols and arcs for the control of nonterminal occurrences in sentential forms, Marek and Češka showed that for every random-context grammar, an isomorphic Petri net can be constructed, where each derivation of the grammar is simulated by some occurrence sequence of transitions of the Petri net, and vice versa. In [39] the relationship between vector grammars and Petri nets was investigated, partially, hybrid Petri nets were introduced and the equality of the family of hybrid Petri net languages and the family of vector languages was shown. By reduction to Petri net reachability problems, Hauschildt and Jantzen [35] could solve a number of open problems in regulated rewriting systems, specifically, every matrix language without appearance checking over one letter alphabet is regular and the finiteness problem for the families of matrix and random context languages is decidable; In several papers [2, 14, 25], Petri nets are used as minimization techniques for context-free (graph) grammars. For instance, in [2], algorithms to eliminate erasing and unit (chain) rules, algorithms to remove useless rules using the Petri net concept are introduced.

Control by Petri nets has also been introduced and studied in automata theory [26–28, 38] and grammar systems theory [6].

In this chapter we summarize the recent obtained results on Petri net controlled grammars and propose new problems for further research.

In Section 2 we recall some basic concepts and results from the areas formal languages and Petri nets: strings, grammars, languages, Petri nets, Petri net languages and so on, which will be used in the next sections.

In Section 3 we define a *context-free Petri net* (a *cf Petri net* for short), where places correspond to nonterminals, transitions are the counterpart of the production rules, the tokens reflect the occurrences of symbols in the sentential form, and there is a one-to-one correspondence between the application of (sequence of) rules and the firing of (sequence of) transitions. Further, we introduce grammars controlled by k-Petri nets, i.e., cf Petri nets with additional k places, and studies the computational power and closure properties of families of languages generated by k-Petri net controlled grammars.

In Section 4 we consider a generalization of the k-Petri net controlled grammars: we associate an arbitrary place/ transition net with a context-free grammar and require that the sequence of applied rules corresponds to an occurrence sequence of transitions in the Petri net. With respect to different labeling strategies and different definitions of final marking sets, we define various classes of Petri net controlled grammars. Here we study the influence of the labeling functions and the effect of the final markings on the generative power.

It is known that many decision problems in formal language theory are equivalent to the reachability problem in Petri net theory, which has been shown that it is decidable, however, it has exponential time complexity. The result of this has been the definition of a number of structural subclasses of Petri nets with a smaller complexity and still adequate modeling power. Thus, it is interesting to consider grammars controlled by such kind of subclasses of Petri nets. In Section 5 we continue our study of arbitrary Petri net controlled grammars by restricting Petri nets to their structural subclasses, i.e., special Petri nets such as state machines, marked graphs, and free-choice nets, and so on.

In Section 6 we examine Petri net controlled grammars with respect to dynamical properties of Petri nets: we use (cf and arbitrary) Petri nets with place capacities. We also investigate capacity-bounded grammars which are counterparts of grammars controlled by Petri nets with place capacities.

In Section 7 we draw some general conclusions and present suggestions for further research.

2. Preliminaries

In this section we recall some prerequisites, by giving basic notions and notations of the theories formal languages, Petri nets and Petri net languages which are used in the next sections. The reader is referred to [16, 34, 36, 42, 45, 47, 50, 52] for further information.

2.1 General notions and notations

Throughout the chapter we use the following general notations. $\in$ denotes the membership of an element to a set while the negation of set membership is denoted by $\notin$. The inclusion

is denoted by $\subseteq$ and the strict (proper) inclusion is denoted by $\subset$. The symbol $\emptyset$ denotes the empty set. The set of positive (non-negative) integers is denoted by $\mathbb{N}$ ($\mathbb{N}_0$). The set of integers is denoted by $\mathbb{Z}$. The power set of a set X is denoted by 2^X, while the cardinality of a set X is denoted by $|X|$.

Let Σ be an *alphabet* which is a finite nonempty set of symbols. A *string* (sometimes a *word*) over the alphabet Σ is a finite sequence of symbols from Σ. The *empty* string is denoted by λ. The *length* of a word w, denoted by $|w|$, is the number of occurrences of symbols in w. The number of occurrences of a symbol a in a string w is denoted by $|w|_a$. The set of all strings over the alphabet Σ is denoted by Σ^*. The set of nonempty strings over Σ is denoted by Σ^+, i.e., $\Sigma^+ = \Sigma^* - \{\lambda\}$. A subset of Σ^* is called a *language*. A language $L \in \Sigma^*$ is *λ-free* if $\lambda \notin L$. For two languages $L_1, L_2 \subseteq \Sigma^*$ the *operation shuffle* is defined by

$$\begin{aligned}\mathrm{Shuf}(L_1, L_2) = \{&u_1v_1u_2v_2 \cdots u_nv_n \mid u_1u_2 \cdots u_n \in L_1, v_1v_2 \cdots v_n \in L_2,\\ &u_i, v_i \in \Sigma^*, 1 \leq i \leq n\}\end{aligned}$$

and for $L \subseteq \Sigma^*$, $\mathrm{Shuf}^*(L) = \bigcup_{k\geq 1} \mathrm{Shuf}^k(L)$ where

$$\mathrm{Shuf}^1(L) = L \text{ and } \mathrm{Shuf}^k(L) = \mathrm{Shuf}(\mathrm{Shuf}^{k-1}(L), L), k \geq 2.$$

2.2 Grammars

A *phrase structure* (*Chomsky*) grammar is a quadruple $G = (V, \Sigma, S, R)$ where V and Σ are two disjoint alphabets of *nonterminal* and *terminal* symbols, respectively, $S \in V$ is the *start symbol* and $R \subseteq (V \cup \Sigma)^* V (V \cup \Sigma)^* \times (V \cup \Sigma)^*$ is a finite set of (*production*) *rules*. Usually, a rule $(u, v) \in R$ is written in the form $u \to v$. A rule of the form $u \to \lambda$ is called an *erasing rule*.

A phrase structure grammar $G = (V, \Sigma, S, R)$ is called a *GS grammar* (a phrase structure grammar due to Ginsburg and Spanier [31]) if $R \subseteq V^+ \times (V \cup \Sigma)^*$.

The families of languages generated by GS grammars and by phrase structure grammars are denoted by **GS** and **RE**, respectively. It is well-known that the family **GS** is equal to the family **RE**.

A string $x \in (V \cup \Sigma)^*$ *directly derives* a string $y \in (V \cup \Sigma)^*$ in G, written as $x \Rightarrow y$ if and only if there is a rule $u \to v \in R$ such that $x = x_1ux_2$ and $y = x_1vx_2$ for some $x_1, x_2 \in (V \cup \Sigma)^*$. The reflexive and transitive closure of the relation $\Rightarrow$ is denoted by $\Rightarrow^*$. A derivation using the sequence of rules $\pi = r_1r_2 \cdots r_k$, $r_i \in R$, $1 \leq i \leq k$, is denoted by $\stackrel{\pi}{\Rightarrow}$ or $\stackrel{r_1r_2\cdots r_k}{\Longrightarrow}$. The *language* generated by G, denoted by $L(G)$, is defined by $L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$.

A phrase-structure grammar $G = (V, \Sigma, S, R)$ is called *context-sensitive* if each rule $u \to v \in R$ has $u = u_1Au_2$, $v = u_1xu_2$ for $u_1, u_2 \in (V \cup \Sigma)^*$, $A \in V$ and $x \in (V \cup \Sigma)^+$ (in context sensitive grammars $S \to \lambda$ is allowed, provided that S does not appear in the right-hand members of rules in R); *context-free* if each rule $u \to v \in R$ has $u \in V$; *linear* if each rule $u \to v \in R$ has $u \in V$ and $v \in \Sigma^* \cup \Sigma^* V \Sigma^*$; *regular* if each rule $u \to v \in R$ has $u \in V$ and $v \in \Sigma \cup \Sigma V$.

The families of languages generated by context-sensitive, context-free, linear and regular grammars are denoted by **CS**, **CF**, **LIN** and **REG**, respectively. Further we denote the family of finite languages by **FIN**. The next strict inclusions, named *Chomsky hierarchy*, hold (for details, see [52]):

Theorem 1. **FIN** $\subset$ **REG** $\subset$ **LIN** $\subset$ **CF** $\subset$ **CS** $\subset$ **RE**.

2.3 Regulated grammars

The idea of regulated rewriting consists of restricting the application of the rules in a context-free grammar in order to avoid some derivations and hence obtaining a subset of the context-free language generated in usual way. The computational power of some context-free grammars with regulated rewriting turns out to be greater than the power of context-free grammars.

A *regularly controlled grammar* is a quintuple $G = (V, \Sigma, S, R, K)$ where V, Σ, S, R are specified as in a context-free grammar and K is a regular set over R. The language generated by G consists of all words $w \in \Sigma^*$ such that there is a derivation $S \xRightarrow{r_1 r_2 \cdots r_n} w$ where $r_1 r_2 \cdots r_n \in K$.

A *matrix grammar* is a quadruple $G = (V, \Sigma, S, M)$ where V, Σ, S are defined as for a context-free grammar, M is a finite set of *matrices* which are finite strings over a set R of context-free rules (or finite sequences of context-free rules). The language generated by the grammar G is $L(G) = \{w \in \Sigma^* \mid S \overset{\pi}{\Rightarrow} w \text{ and } \pi \in M^*\}$.

A *vector grammar* is a quadruple $G = (V, \Sigma, S, M)$ whose components are defined as for a matrix grammar. The language generated by the grammar G is defined by

$$L(G) = \{w \in \Sigma^* \mid S \overset{\pi}{\Rightarrow} w \text{ and } \pi \in \mathrm{Shuf}^*(M)\}.$$

An *additive valence grammar* is a quintuple $G = (V, \Sigma, S, R, v)$ where V, Σ, S, R are defined as for a context-free grammar and v is a mapping from R into $\mathbb{Z}$. The language generated by G consists of all strings $w \in \Sigma^*$ such that there is a derivation $S \xRightarrow{r_1 r_2 \cdots r_n} w$ where $\sum_{i=1}^{n} v(r_i) = 0$.

A *positive valence grammar* is a quintuple $G = (V, \Sigma, S, R, v)$ whose components are defined as for additive valence grammars. The language generated by G consists of all strings $w \in \Sigma^*$ such that there is a derivation $S \xRightarrow{r_1 r_2 \cdots r_n} w$ where $\sum_{i=1}^{n} v(r_i) = 0$ and for any $1 \leq j < n$, $\sum_{i=1}^{j} v(r_i) \geq 0$.

The families of languages generated by regularly controlled, matrix, vector, additive valence and positive valence grammars (with erasing rules) are denoted by **rC**, **MAT**, **VEC**, **aV**, **pV** (**rC**$^\lambda$, **MAT**$^\lambda$, **VEC**$^\lambda$, **aV**$^\lambda$, **pV**$^\lambda$), respectively.

Theorem 2. *The following inclusions and equalities hold (for details, see [16]):*

(1) $\mathbf{CF} \subset \mathbf{aV} = \mathbf{aV}^\lambda \subset \mathbf{MAT} = \mathbf{rC} = \mathbf{pV}$;

(2) $\mathbf{MAT} \subseteq \mathbf{VEC} \subset \mathbf{CS}$;

(3) $\mathbf{MAT} \subseteq \mathbf{MAT}^\lambda = \mathbf{rC}^\lambda = \mathbf{VEC}^\lambda = \mathbf{pV}^\lambda \subset \mathbf{RE}$.

2.4 Petri nets

A *Petri net* (PN) is a construct $N = (P, T, F, \phi)$ where P and T are disjoint finite sets of *places* and *transitions*, respectively, $F \subseteq (P \times T) \cup (T \times P)$ is the set of *directed arcs*, $\phi : F \to \mathbb{N}$ is a *weight function*.

A Petri net can be represented by a bipartite directed graph with the node set $P \cup T$ where places are drawn as *circles*, transitions as *boxes* and arcs as *arrows*. The arrow representing an arc $(x, y) \in F$ is labeled with $\phi(x, y)$; if $\phi(x, y) = 1$, then the label is omitted.

An *ordinary net* (ON) is a Petri net $N = (P, T, F, \phi)$ where $\phi(x, y) = 1$ for all $(x, y) \in F$. We omit ϕ from the definition of an ordinary net, i.e., $N = (P, T, F)$.

A mapping $\mu : P \to \mathbb{N}_0$ is called a *marking*. For each place $p \in P$, $\mu(p)$ gives the number of *tokens* in p. Graphically, tokens are drawn as small solid *dots* inside circles. ${}^\bullet x = \{y \mid (y, x) \in F\}$ and $x^\bullet = \{y \mid (x, y) \in F\}$ are called *pre-* and *post-sets* of $x \in P \cup T$, respectively. For $t \in T$ ($p \in P$), the elements of ${}^\bullet t$ (${}^\bullet p$) are called *input* places (transitions) and the elements of $t^\bullet$ ($p^\bullet$) are called *output* places (transitions) of t (p).

A transition $t \in T$ is *enabled* by marking μ if and only if $\mu(p) \geq \phi(p, t)$ for all $p \in {}^\bullet t$. In this case t can *occur* (*fire*). Its occurrence transforms the marking μ into the marking μ' defined for each place $p \in P$ by $\mu'(p) = \mu(p) - \phi(p, t) + \phi(t, p)$. We write $\mu \xrightarrow{t} \mu'$ to indicate that the firing of t in μ leads to μ'. A marking μ is called *terminal* if in which no transition is enabled. A finite sequence $t_1 t_2 \cdots t_k$, $t_i \in T, 1 \leq i \leq k$, is called *an occurrence sequence* enabled at a marking μ and finished at a marking μ_k if there are markings $\mu_1, \mu_2, \ldots, \mu_{k-1}$ such that $\mu \xrightarrow{t_1} \mu_1 \xrightarrow{t_2} \ldots \xrightarrow{t_{k-1}} \mu_{k-1} \xrightarrow{t_k} \mu_k$. In short this sequence can be written as $\mu \xrightarrow{t_1 t_2 \cdots t_k} \mu_k$ or $\mu \xrightarrow{\nu} \mu_k$ where $\nu = t_1 t_2 \cdots t_k$. For each $1 \leq i \leq k$, marking μ_i is called *reachable* from marking μ. $\mathcal{R}(N, \mu)$ denotes the set of all reachable markings from a marking μ.

A *marked* Petri net is a system $N = (P, T, F, \phi, \iota)$ where (P, T, F, ϕ) is a Petri net, ι is the *initial marking*.

A Petri net *with final markings* is a construct $N = (P, T, F, \phi, \iota, M)$ where (P, T, F, ϕ, ι) is a marked Petri net and $M \subseteq \mathcal{R}(N, \iota)$ is set of markings which are called *final* markings. An occurrence sequence ν of transitions is called *successful* for M if it is enabled at the initial marking ι and finished at a final marking τ of M. If M is understood from the context, we say that ν is a successful occurrence sequence.

A Petri net N is said to be *k-bounded* if the number of tokens in each place does not exceed a finite number k for any marking reachable from the initial marking ι, i.e., $\mu(p) \leq k$ for all $p \in P$ and for all $\mu \in \mathcal{R}(N, \iota)$. A Petri net N is said to be *bounded* if it is k-bounded for some $k \geq 1$.

A Petri net with *place capacity* is a system $N = (P, T, F, \phi, \iota, \kappa)$ where (P, T, F, ϕ, ι) is a marked Petri net and $\kappa : P \to \mathbb{N}_0$ is a function assigning to each place a number of maximal admissible tokens. A marking μ of the net N is valid if $\mu(p) \leq \kappa(p)$, for each place $p \in P$. A transition $t \in T$ is *enabled* by a marking μ if additionally the successor marking is valid.

2.5 Special Petri nets

It is known that many decision problems are equivalent to the reachability problem [33], which has been shown to be decidable. However, it has exponential space complexity [40], thus from a practical point of view, Petri nets may be too powerful to be analyzed. The result of this has been the definition of a number of subclasses of Petri nets in order to find a subclass with a smaller complexity and still adequate modeling power for practical purposes. These subclasses are defined by restrictions on their structure intended to improve their analyzability. We consider the following main structural subclasses of Petri nets.

A *state machine* (SM) is an ordinary Petri net such that each transition has exactly one input place and exactly one output place, i.e., $|{}^\bullet t| = |t^\bullet| = 1$ for all $t \in T$. This means that there can not be concurrency but there can be conflict.

A *generalized state machine* (GSM) is an ordinary Petri net such that $|{}^\bullet t| \leq 1$ and $|t^\bullet| \leq 1$ for all $t \in T$.

A *marked graph* (MG) is an ordinary Petri net such that each place has exactly one input transition and exactly one output transition, i.e., $|{}^\bullet p| = |p^\bullet| = 1$ for all $p \in P$. This means that there can not be conflict but there can be concurrency.

A *generalized marked graph* (GMG) is an ordinary Petri net such that $|{}^\bullet p| \leq 1$ and $|p^\bullet| \leq 1$ for all $p \in P$.

A *casual net* (CN) is a generalized marked graph each subgraph of which is not a a cycle.

A *free-choice* net (FC) is an ordinary Petri net such every arc is either the only arc going from the place, or it is the only arc going to a transition, i.e., that if $p_1^\bullet \cap p_2^\bullet \neq \emptyset$ then $|p_1^\bullet| = |p_2^\bullet| = 1$ for all $p_1, p_2 \in P$. This means that there can be both concurrency and conflict but not the same time.

An *extended free-choice* net (EFC) is an ordinary Petri net such that if $p_1^\bullet \cap p_2^\bullet \neq \emptyset$ then $p_1^\bullet = p_2^\bullet$ for all $p_1, p_2 \in P$.

An *asymmetric choice net* (AC) is an ordinary Petri net such that if $p_1^\bullet \cap p_2^\bullet \neq \emptyset$ then $p_1^\bullet \subseteq p_2^\bullet$ or $p_1^\bullet \supseteq p_2^\bullet$ for all $p_1, p_2 \in P$. In asymmetric choice nets concurrency and conflict (in sum, confusion) may occur but not asymmetrically.

3. k-Petri net controlled grammars

Since a context-free grammar and its derivation process can also be described by a Petri net (see [11]), where places correspond to nonterminals, transitions are the counterpart of the production rules, and the tokens reflect the occurrences of symbols in the sentential form, and there is a one-to-one correspondence between the application of (sequence of) rules and the firing of (sequence of) transitions, it is a very natural and very easy idea to control the derivations in a context-free grammar by adding some features to the associated Petri net. In this section we introduce a Petri net associated with a context-free grammar (i.e., a *context-free Petri net*), construct Petri net control mechanisms from cf Petri nets by adding new places, and define the corresponding grammars, called *k-Petri net controlled grammars*.

The construction of the following type of Petri nets is based on the idea of using similarity between the firing of a transition and the application of a production rule in a derivation in which places are nonterminals and tokens are separate occurrences of nonterminals.

Definition 1. A *context-free Petri net* (in short, a *cf Petri net*) w.r.t. a context-free grammar $G = (V, \Sigma, S, R)$ is a septuple $N = (P, T, F, \phi, \beta, \gamma, \iota)$ where

- (P, T, F, ϕ) is a Petri net;
- labeling functions $\beta : P \to V$ and $\gamma : T \to R$ are bijections;
- there is an arc from place p to transition t if and only if $\gamma(t) = A \to \alpha$ and $\beta(p) = A$. The weight of the arc (p, t) is 1;

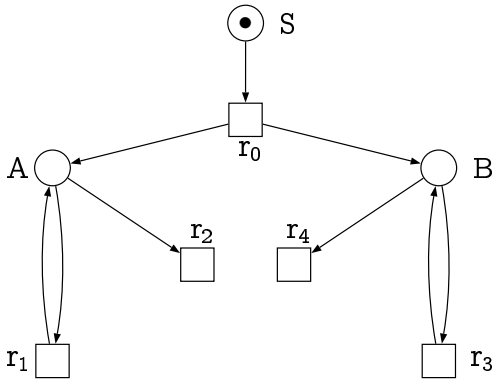

Figure 1. A cf Petri net N_1

- there is an arc from transition t to place p if and only if $\gamma(t) = A \to \alpha$ and $\beta(p) = x$ where $|\alpha|_x > 0$. The weight of the arc (t, p) is $|\alpha|_x$;
- the initial marking ι is defined by $\iota(\beta^{-1}(S)) = 1$ and $\iota(p) = 0$ for all $p \in P - \{\beta^{-1}(S)\}$.

Example 1. Let G_1 be a context-free grammar with the rules:

$$r_0 : S \to AB,\ r_1 : A \to aAb,\ r_2 : A \to ab,\ r_3 : B \to cB,\ r_4 : B \to c$$

(the other components of the grammar can be seen from these rules). Figure 1 illustrates a cf Petri net N_1 with respect to the grammar G_1. Obviously, $L(G_1) = \{a^n b^n c^m \mid n, m \geq 1\}$.

The following proposition shows the similarity between terminal derivations in a context-free grammar and successful occurrences of transitions in the corresponding cf Petri net.

Proposition 3. *Let $N = (P, T, F, \phi, \iota, \beta, \gamma)$ be the cf Petri net with respect to a context-free grammar $G = (V, \Sigma, S, R)$. Then $S \xRightarrow{r_1 r_2 \cdots r_n} w$, $w \in \Sigma^*$ is a derivation in G iff $t_1 t_2 \cdots t_n$, $\iota \xrightarrow{t_1 t_2 \cdots t_n} \mu_n$, is an occurrence sequence of transitions in N such that $\gamma(t_1 t_2 \cdots t_n) = r_1 r_2 \cdots r_n$ and $\mu_n(p) = 0$ for all $p \in P$.*

Now we define a *k-Petri net*, i.e., a cf Petri net with additional k places and additional arcs from/to these places to/from transitions of the net, the pre-sets and post-sets of the additional places are disjoint.

Definition 2. Let $G = (V, \Sigma, S, R)$ be a context-free grammar with its corresponding cf Petri net $N = (P, T, F, \phi, \beta, \gamma, \iota)$. Let k be a positive integer and let $Q = \{q_1, q_2, \ldots, q_k\}$ be a set of new places called *counters*. A *k-Petri net* is a construct $N_k = (P \cup Q, T, F \cup E, \varphi, \zeta, \gamma, \mu_0, \tau)$ where

- $E = \{(t, q_i) \mid t \in T_1^i, 1 \leq i \leq k\} \cup \{(q_i, t) \mid t \in T_2^i, 1 \leq i \leq k\}$ such that $T_1^i \subset T$ and $T_2^i \subset T$, $1 \leq i \leq k$ where $T_l^i \cap T_l^j = \emptyset$ for $1 \leq l \leq 2$, $T_1^i \cap T_2^j = \emptyset$ for $1 \leq i < j \leq k$ and $T_1^i = \emptyset$ if and only if $T_2^i = \emptyset$ for any $1 \leq i \leq k$.
- the weight function $\varphi(x, y)$ is defined by $\varphi(x, y) = \phi(x, y)$ if $(x, y) \in F$ and $\varphi(x, y) = 1$ if $(x, y) \in E$,

- the labeling function $\zeta : P \cup Q \to V \cup \{\lambda\}$ is defined by $\zeta(p) = \beta(p)$ if $p \in P$ and $\zeta(p) = \lambda$ if $p \in Q$,
- the initial marking μ_0 is defined by $\mu_0(\beta^{-1}(S)) = 1$ and $\mu_0(p) = 0$ for all $p \in P \cup Q - \{\beta^{-1}(S)\}$,
- τ is the final marking where $\tau(p) = 0$ for all $p \in P \cup Q$.

Definition 3. A *k-Petri net controlled grammar* (*k-PN controlled grammar* for short) is a quintuple $G = (V, \Sigma, S, R, N_k)$ where V, Σ, S, R are defined as for a context-free grammar and N_k is a k-PN with respect to the context-free grammar (V, Σ, S, R).

Definition 4. The *language* generated by a k-Petri net controlled grammar G consists of all strings $w \in \Sigma^*$ such that there is a derivation

$$S \xRightarrow{r_1 r_2 \cdots r_n} w \text{ where } t_1 t_2 \cdots t_n = \gamma^{-1}(r_1 r_2 \cdots r_n) \in T^*$$

is an occurrence sequence of the transitions of N_k enabled at the initial marking ι and finished at the final marking τ.

We denote the family of languages generated by k-PN controlled grammars (with erasing rules) by $\mathbf{PN}_k$ ($\mathbf{PN}_k^{\lambda}$), $k \geq 1$.

Example 2. Let G_2 be a 2-PN controlled grammar with the production rules:

$$\begin{array}{lll} r_0 : S \to A_1 B_1 A_2 B_2, & r_1 : A_1 \to a_1 A_1 b_1, & r_2 : A_1 \to a_1 b_1, \\ r_3 : B_1 \to c_1 B_1, & r_4 : B_1 \to c_1, & r_5 : A_2 \to a_2 A_2 b_2, \\ r_6 : A_2 \to a_2 b_2, & r_7 : B_2 \to c_2 B_2, & r_8 : B_2 \to c_2 \end{array}$$

and the corresponding 2-Petri net N_2 is given in Figure 2. Then it is easy to see that G_2 generates the language

$$L(G_2) = \{a_1^n b_1^n c_1^n a_2^m b_2^m c_2^m \mid n, m \geq 1\}.$$

Theorem 4. *The language*

$$L = \prod_{i=1}^{k+1} a_i^{n_i} b_i^{n_i} c_i^{n_i}$$

where $k \geq 1$ and $n_i \geq 1$, $1 \leq i \leq k+1$, cannot be generated by a k-PN controlled grammar.

The following theorem presents the relations of languages generated by k-Petri net controlled grammars to context-free, (positive) additive valence and vector languages.

Theorem 5.

$$\mathbf{CF} \subset \mathbf{PN}_1^{[\lambda]} \subseteq \mathbf{pV}^{[\lambda]}, \ \mathbf{aV}^{[\lambda]} \subset \mathbf{PN}_2^{[\lambda]} \text{ and } \mathbf{PN}_n^{[\lambda]} \subseteq \mathbf{VEC}^{[\lambda]}, n \geq 1.$$

The next theorem shows that the language families generated by k-Petri net controlled grammars form infinite hierarchy with respect to the numbers of additional places.

Theorem 6. *For $k \geq 1$, $\mathbf{PN}_k^{[\lambda]} \subset \mathbf{PN}_{k+1}^{[\lambda]}$.*

The closure properties of the language families generated by k-PN controlled grammars are given in the following theorem.

Theorem 7. *The family of languages $\mathbf{PN}_k$, $k \geq 1$, is closed under union, substitution, mirror image, intersection with regular languages and it is not closed under concatenation.*

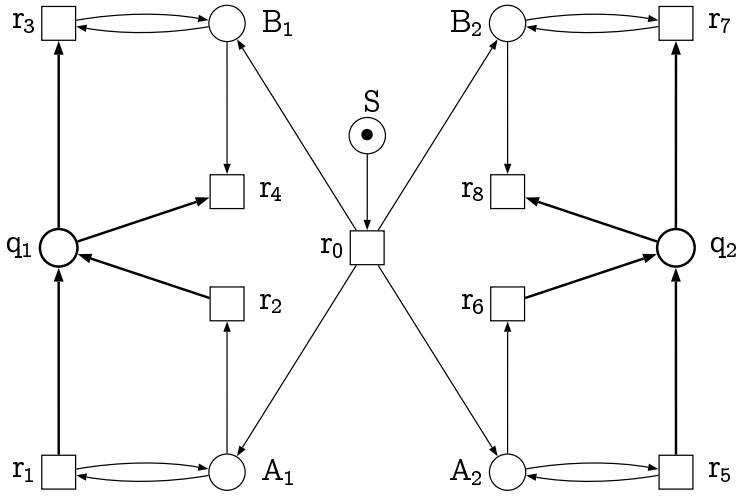

Figure 2. A 2-Petri net N_2

4. Arbitrary Petri net controlled grammars

In this section we consider a generalization of regularly controlled grammars: instead of a finite automaton we associate a Petri net with a context-free grammar and require that the sequence of applied rules corresponds to an occurrence sequence of the Petri net, i.e., to sequences of transitions which can be fired in succession. However, one has to decide what type of correspondence is used and what concept is taken as an equivalent of acceptance. Since the sets of occurrence sequences form the language of a Petri net, we choose the correspondence and the equivalent for acceptance according to the variations which are used in the theory of Petri net languages.

Therefore as correspondence we choose a bijection (between transitions and rules) or a coding (any transition is mapped to a rule) or a weak coding (any transition is mapped to a rule or the empty word) which agree with the classical three variants of Petri net languages (see e.g. [34, 57, 58]).

We consider two types of acceptance from the theory of Petri net languages: only those occurrence sequences belonging to the languages which transform the initial marking into a marking from a given finite set of markings or all occurrence sequences are taken (independent of the obtained marking). If we use only the occurrence sequence leading to a marking in a given finite set of markings we say that the Petri net controlled grammar is of t-type; if we consider all occurrence sequences, then the grammar is of r-type. We add a further type which can be considered as a complement of the t-type. Obviously, if we choose a finite set M of markings and require that the marking obtained after the application of the occurrence sequence is smaller than at least one marking of M (the order is componentwise), then we can choose another finite set M' of markings and require that the obtained marking belongs to M'. The complementary approach requires that the obtained marking is larger than at least one marking of the given set M. The corresponding class of Petri net controlled grammars is called of g-type. Therefore, we obtain nine classes of Petri net controlled

grammars since we have three different types of correspondence and three types of the set of admitted occurrence sequences. These types of control are generalizations of those types of control considered in the previous chapter, too, where instead of arbitrary Petri nets only such Petri nets have been considered where the places and transitions correspond in a one-to-one manner to nonterminals and rules, respectively.

We now introduce the concept of control by an arbitrary Petri net.

Definition 5. An *arbitrary Petri net controlled grammar* is a tuple $G = (V, \Sigma, S, R, N, \gamma, M)$ where V, Σ, S, R are defined as for a context-free grammar and $N = (P, T, F, \varphi, \iota)$ is a (marked) Petri net, $\gamma : T \to R \cup \{\lambda\}$ is a transition labeling function and M is a set of final markings.

Definition 6. The *language* generated by a Petri net controlled grammar G, denoted by $L(G)$, consists of all strings $w \in \Sigma^*$ such that there is a derivation $S \overset{r_1 r_2 \cdots r_k}{\Longrightarrow} w \in \Sigma^*$ and an occurrence sequence $\nu = t_1 t_2 \cdots t_s$ which is successful for M such that $r_1 r_2 \cdots r_k = \gamma(t_1 t_2 \cdots t_s)$.

Example 3. Let $G_3 = (\{S, A, B, C\}, \{a, b, c\}, S, R, N_3, \gamma, M)$ be a Petri net controlled grammar where R consists of

$$S \to ABC,$$

$$A \to aA, \qquad B \to bB, \qquad C \to cC,$$

$$A \to a, \qquad B \to b, \qquad C \to c$$

and N_3 is illustrated in Figure 3. If M is the set of all reachable markings, then G_3 generates the language

$$L(G_3) = \{a^n b^m c^k \mid n \geq m \geq k \geq 1\}.$$

If $M = \{\mu\}$ with $\mu(p) = 0$ for all $p \in P$, then it generates the language

$$L(G_3) = \{a^n b^n c^n \mid n \geq 1\}.$$

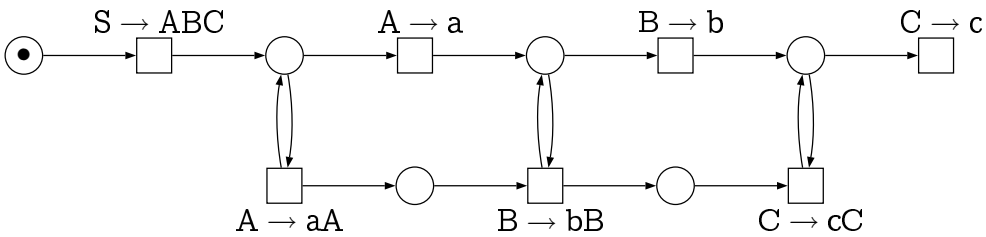

Figure 3. A labeled Petri net N_3

Different labeling strategies and different definitions of the set of final markings result in various types of Petri net controlled grammars. We consider the following types of Petri net controlled grammars.

Definition 7. A Petri net controlled grammar $G = (V, \Sigma, S, R, N, \gamma, M)$ is called *free* (abbreviated by f) if a different label is associated to each transition, and no transition is labeled with the empty string; *λ-free* (abbreviated by $-\lambda$) if no transition is labeled with the empty string; *extended* (abbreviated by λ) if no restriction is posed on the labeling function γ.

Definition 8. A Petri net controlled grammar $G = (V, \Sigma, S, R, N, \gamma, M)$ is called *r-type* if M is the set of all reachable markings from the initial marking ι, i.e., $M = \mathcal{R}(N, \iota)$; *t-type* if $M \subseteq \mathcal{R}(N, \iota)$ is a finite set; *g-type* if for a given finite set $M_0 \subseteq \mathcal{R}(N, \iota)$, M is the set of all markings such that for every marking $\mu \in M$ there is a marking $\mu' \in M_0$ such that $\mu \geq \mu'$.

We use the notation (x, y)-*PN controlled grammar* where $x \in \{f, -\lambda, \lambda\}$ shows the type of a labeling function and $y \in \{r, t, g\}$ shows the type of a set of final markings. We denote by $\mathbf{PN}(x, y)$ and $\mathbf{PN}^{\lambda}(x, y)$ the families of languages generated by (x, y)-PN controlled grammars without and with erasing rules, respectively, where $x \in \{f, -\lambda, \lambda\}$ and $y \in \{r, t, g\}$.

The following theorem shows that the labeling strategy does not effect on the generative capacity of arbitrary Petri net controlled grammars.

Theorem 8. *For* $y \in \{r, t, g\}$,

$$\mathbf{PN}^{[\lambda]}(f, y) = \mathbf{PN}^{[\lambda]}(-\lambda, y) = \mathbf{PN}^{[\lambda]}(\lambda, y).$$

Not surprisingly, arbitrary Petri net controlled grammars generate matrix languages. Moreover, in [65] it was proven that the erasing rules in arbitrary Petri net controlled grammars can be eliminated without effecting on the generative power of the grammars. If we take into consideration this result, we obtain the following inclusions and equalities:

Theorem 9. *For* $x \in \{f, -\lambda, \lambda\}$ *and* $y \in \{r, t, g\}$,

$$\mathbf{MAT} \subseteq \mathbf{PN}(x, r) = \mathbf{PN}(x, g) \subseteq \mathbf{PN}(x, t) = \mathbf{PN}^{\lambda}(x, y) = \mathbf{MAT}^{\lambda}.$$

5. Grammars controlled by special Petri nets

In the previous section we investigated arbitrary Petri net controlled grammars in dependence on the type of labeling functions and on the definitions of final markings, and showed that Petri net controlled grammars have the same power as some other regulating mechanisms such as matrices, finite automata. If we consider these matrices and finite automata in terms of control mechanisms, special types of matrices and special regular languages are widely investigated in literature, for instance, as control, simple matrices ([37]) or some subclasses of regular languages ([15, 17]) are considered. Thus, it is also natural to investigate grammars controlled by some special classes of Petri nets. We consider (generalized) state machines, (generalized) marked graphs, causal nets, (extended) free-choice nets, asymmetric choice nets and ordinary nets. Similarly to the general case we also investigate the effects of labeling policies and final markings to the computational power, and prove that the family of languages generated by (arbitrary) Petri net controlled grammars coincide with the family of languages generated by grammars controlled by free-choice nets.

Let $G = (V, \Sigma, S, R, N, \gamma, M)$ be an arbitrary Petri net controlled grammar. The grammar G is called a (generalized) state machine, (generalized) marked graph, causal net, (extended) free-choice net, asymmetric choice net or ordinary net controlled grammar if the net N is a (generalized) state machine, (generalized) marked graph, causal net, (extended) free-choice net, asymmetric choice net or ordinary net, respectively.

We also use a notation an (x, y)-(generalized) state machine, ((generalized) marked graph, causal net, (extended) free-choice net, asymmetric choice net and ordinary net) controlled grammar where $x \in \{f, -\lambda, \lambda\}$ shows the type of a labeling function γ and $y \in \{r, t, g\}$ shows the type of a set of final markings.

We denote the families of languages generated by grammars controlled by state machines, generalized state machines, marked graphs, generalize marked graphs, causal nets, free-choice nets, extended free-choice nets, asymmetric nets, ordinary nets and Petri nets

$$\mathbf{SM}^{[\lambda]}(x, y),\ \mathbf{GSM}^{[\lambda]}(x, y),\ \mathbf{MG}^{[\lambda]}(x, y),\ \mathbf{GMG}^{[\lambda]}(x, y),\ \mathbf{CN}^{[\lambda]}(x, y),$$
$$\mathbf{FC}^{[\lambda]}(x, y),\ \mathbf{EFC}^{[\lambda]}(x, y),\ \mathbf{AC}^{[\lambda]}(x, y),\ \mathbf{ON}^{[\lambda]}(x, y),\ \mathbf{PN}^{[\lambda]}(x, y)$$

where $x \in \{f, -\lambda, \lambda\}$ and $y \in \{r, t, g\}$.

The inclusion $\mathbf{X}(x, y) \subseteq \mathbf{X}^{\lambda}(x, y)$ immediately follows from the definition where

- $\mathbf{X} \in \{\mathbf{SM}, \mathbf{GSM}, \mathbf{MG}, \mathbf{GMG}, \mathbf{CN}, \mathbf{FC}, \mathbf{EFC}, \mathbf{AC}, \mathbf{ON}\}$,
- $x \in \{f, -\lambda, \lambda\}$ and $y \in \{r, t, g\}$.

Example 4. Let $G_4 = (\{S, A, B\}, \{a, b\}, S, R, N_4, \gamma, M)$ be a SM controlled grammar where R consists of

$$S \to AB,$$
$$A \to aA, \quad A \to bA, \quad A \to \lambda,$$
$$B \to aB, \quad B \to bB, \quad B \to \lambda,$$

the Petri net N_4 illustrated in Figure 4 is a labeled state machine and $M = \{\mu\}$ where $\mu(p_0) = 1$ and $\mu(p) = 0$ for all $p \in P - \{p_0\}$, then

$$L(G_4) = \{ww \mid w \in \{a, b\}^*\} \in \mathbf{SM}^{\lambda}(\lambda, t).$$

Example 5. Let $G_5 = (\{S, A, B\}, \{a, b\}, S, R, N_5, \gamma', M')$ be a MG controlled grammar where R is as for the grammar G_1 in Example 4, a labeled marked graph N_5 is illustrated in Figure 5 and $M' = \{\mu\}$ where $\mu(p) = 0$ for all $p \in P$. Then

$$L(G_5) = \{ww' \mid w \in \{a, b\}^* \text{ and } w' \in \mathrm{Perm}(w)\} \in \mathbf{MG}^{\lambda}(\lambda, t).$$

We have the same result on the labeling strategies as that in the previous section: the labeling of transitions of special Petri nets do not effect on the generative powers of the families of languages generated by grammars controlled by these nets.

Theorem 10. *For* $\mathbf{X} \in \{\mathbf{SM}, \mathbf{GSM}, \mathbf{MG}, \mathbf{GMG}, \mathbf{CN}, \mathbf{FC}, \mathbf{EFC}, \mathbf{AC}, \mathbf{ON}\}$, *and* $y \in \{r, t, g\}$,

$$\mathbf{X}^{[\lambda]}(f, y) = \mathbf{X}^{[\lambda]}(-\lambda, y) = \mathbf{X}^{[\lambda]}(\lambda, y).$$

The following theorem shows the relations of families of languages generated by special Petri net controlled grammars.

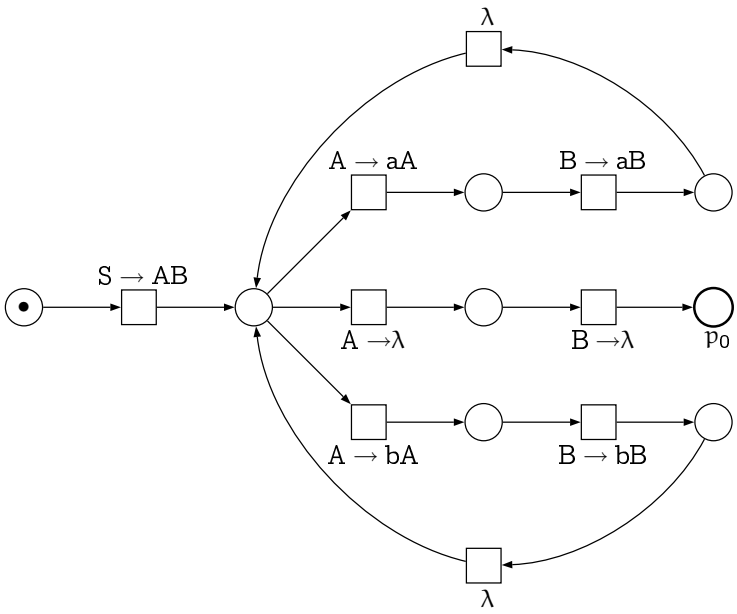

Figure 4. A labeled state machine N_4

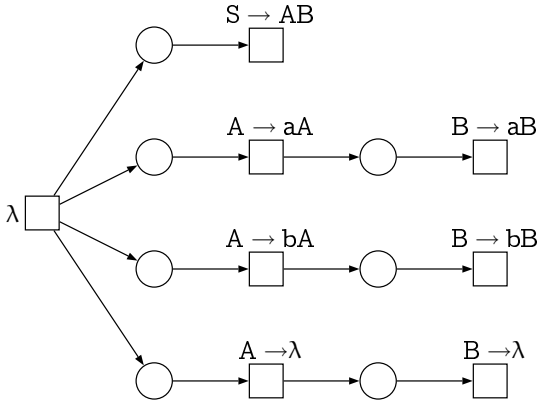

Figure 5. A labeled marked graph N_5

Theorem 11. *The following inclusions and equalities hold:*

$$(1)\ \mathbf{MG}^{[\lambda]}(x,y) = \mathbf{GMG}^{[\lambda]}(x,y) \text{ and } \mathbf{PN}(x,y') = \mathbf{X}(x,y'),$$

$$(2)\ \mathbf{CN}(x,y') \subseteq \mathbf{MG}(x,y') \subseteq \mathbf{PN}(x,y') \subseteq \mathbf{MAT}^{\lambda},$$

$$(3)\ \mathbf{MAT} \subseteq \mathbf{GSM}(x,y') \subseteq \mathbf{PN}(x,y') \subseteq \mathbf{MAT}^{\lambda},$$

$$(4)\ \mathbf{CF} \subset \mathbf{MAT} = \mathbf{SM}(x,y) \subseteq \mathbf{VEC} \subseteq \begin{cases} \mathbf{GSM}(x,t) \\ \mathbf{CN}(x,t) \subseteq \mathbf{MG}(x,t) \end{cases} \subseteq \mathbf{MAT}^{\lambda},$$

$$(5)\ \mathbf{MAT}^{\lambda} = \mathbf{PN}^{\lambda}(x,y) = \mathbf{PN}(x,t) = \mathbf{X}^{\lambda}(x,t) = \mathbf{Y}^{\lambda}(x,t) = \mathbf{Z}^{\lambda}(x,y),$$

where $x \in \{f, -\lambda, \lambda\}$, $y \in \{r,g,t\}$, $y' \in \{r,g\}$ *and* $\mathbf{X} \in \{\mathbf{FC}, \mathbf{EFC}, \mathbf{AC}, \mathbf{ON}\}$, $\mathbf{Y} \in \{\mathbf{MG}, \mathbf{GMG}, \mathbf{CN}\}$, $\mathbf{Z} \in \{\mathbf{SM}, \mathbf{GSM}, \mathbf{FC}, \mathbf{EFC}, \mathbf{AC}, \mathbf{ON}\}$.

6. Capacity-bounded grammars

In this section we continue the research in this direction by restricting to (context-free, extended or arbitrary) Petri nets with place capacities. Quite obviously, a context-free Petri net with place capacity regulates the defining grammar by permitting only those derivations where the number of each nonterminal in each sentential form is bounded by its capacity. Similar mechanisms have been introduced and investigated by several authors. Grammar with *finite index* (the index of a grammar is the maximal number of nonterminals simultaneously appearing in its complete derivations (considering the most economical derivations for each string)) were first considered by Brainerd [7]. *Nonterminal-bounded* grammars (a grammar a nonterminal-bounded if the total number of nonterminals in every sentential form does not exceed an upper bound) were introduced by Altman and Banerji in [3–5]. A "weak" variant of nonterminal-bounded grammars (only the complete derivations are required to be bounded) were defined by Moriya [44]. Ginsburg and Spanier introduced *derivation-bounded* languages in [32] (all strings which have complete derivation in a grammar G consisting of sentential forms each of which does not contain more than k nonterminals collected in the set $L_k(G)$). There it was shown that grammars regulated in this way generate the family of context-free languages of finite index, even if arbitrary nonterminal strings are allowed as left-hand sides of production rules. Finite index restrictions to regulated grammars have also been investigated [29, 30, 48, 49, 51, 53–55]. There it was shown that the families of most regulated languages are collapse.

In this section we show that capacity-bounded context-free grammars have a larger generative power than context-free grammars of finite index while the family of languages generated by capacity-bounded phrase structure grammars (due to Ginsburg and Spanier) and several families of languages generated by grammars controlled by extended cf Petri nets with place capacities coincide with the family of matrix languages of finite index.

We will now introduce capacity-bounded grammars and show some relations to similar concepts known from the literature.

Definition 9. A *capacity-bounded* grammar is a tuple $G = (V, \Sigma, S, R, \kappa)$ where $G' = (V, \Sigma, S, R)$ is a grammar and $\kappa : V \to \mathbb{N}$ is a capacity function. The derivation relation $\Rightarrow_G$ is defined as $\alpha \Rightarrow_G \beta$ iff $\alpha \Rightarrow_{G'} \beta$ and $|\alpha|_A \leq \kappa(A)$ and $|\beta|_A \leq \kappa(A)$, for all $A \in V$. The language of G is defined as $L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$.

The families of languages generated by capacity-bounded GS grammars and by context-free capacity-bounded grammars are denoted by $\mathbf{GS}_{cb}$ and $\mathbf{CF}_{cb}$, respectively. The capacity function mapping each nonterminal to 1 is denoted by $\mathbf{1}$. The notions of finite index and bounded capacities can be extended to matrix, vector and semi-matrix grammars. The corresponding language families are denoted by

$$\mathbf{MAT}_{fin}^{[\lambda]},\ \mathbf{VEC}_{fin}^{[\lambda]},\ \mathbf{sMAT}_{fin}^{[\lambda]},\ \mathbf{MAT}_{cb}^{[\lambda]},\ \mathbf{VEC}_{cb}^{[\lambda]}.$$

Capacity-bounded grammars are very similar to derivation-bounded grammars, which were studied in [32]. A *derivation-bounded* grammar is a quintuple $G = (V, \Sigma, S, R, k)$ where $G' = (V, \Sigma, S, R)$ is a grammar and $k \in \mathbb{N}$ is a bound on the number of allowed nonterminals. The language of G contains all words $w \in L(G')$ that have a derivation $S \Rightarrow^* w$ such that $|\beta|_V \leq k$, for each sentential form β of the derivation.

Other related concepts are nonterminal-bounded grammars and grammars of finite index. A context-free grammar $G = (V, \Sigma, S, R)$ is *nonterminal-bounded* if $|\beta|_V \leq k$ for some fixed $k \in \mathbb{N}$ and all sentential forms β derivable in G. The *index* of a derivation in G is the maximal number of nonterminal symbols in its sentential forms. G is of *finite index* if every word in $L(G)$ has a derivation of index at most k for some fixed $k \in \mathbb{N}$. The family of context-free languages of finite index is denoted by $\mathbf{CF}_{fin}$.

Note that there is a subtle difference between the first two and the last two concepts. While context-free nonterminal-bounded and finite index grammars are just context-free grammars with a certain structural property (and generate context-free languages by definition), capacity-bounded and derivation-bounded grammars are special cases of *regulated rewriting* (and could therefore generate non-context-free languages). However, it has been shown that the family of derivation bounded languages is equal to $\mathbf{CF}_{fin}$, even if arbitrary grammars due to Ginsburg and Spanier are permitted [32]. We will now give an example of capacity-bounded grammars generating non-context-free languages.

Example 6. Let $G = (\{S, A, B, C, D, E, F\}, \{a, b, c\}, S, R, \mathbf{1})$ be the capacity-bounded grammar where R consists of the rules:

$$\begin{array}{llll} r_1 : S \to ABCD, & r_2 : AB \to aEFb, & r_3 : CD \to cAD, & r_4 : EF \to EC, \\ r_5 : EF \to FC, & r_6 : AD \to FD, & r_7 : AD \to ED, & r_8 : EC \to AB, \\ r_9 : FD \to CD, & r_{10} : FC \to AF, & r_{11} : AF \to \lambda, & r_{12} : ED \to \lambda. \end{array}$$

The possible derivations are exactly those of the form

$$\begin{aligned} S &\overset{r_1}{\Rightarrow} ABCD \\ &\overset{(r_2 r_3 r_4 r_6 r_8 r_9)^n}{\Longrightarrow} a^n ABb^n c^n CD \\ &\overset{r_2 r_3}{\Longrightarrow} a^{n+1} EFb^{n+1} c^{n+1} AD \\ &\overset{r_5 r_7}{\Longrightarrow} a^{n+1} FCb^{n+1} c^{n+1} ED \\ &\overset{r_{10} r_{11} r_{12}}{\Longrightarrow} a^n b^n c^n \end{aligned}$$

(in the last phase, the sequences $r_{10} r_{12} r_{11}$ and $r_{12} r_{10} r_{11}$ could also be applied with the same result). Therefore,

$$L(G) = \{a^n b^n c^n \mid n \geq 1\}.$$

The above example shows that capacity-bounded grammars – in contrast to derivation bounded grammars – can generate non-context-free languages. Moreover, any context-free language generated by a grammar of G of finite index is also generated by the capacity-bounded grammar (G, κ) where κ is capacity function constantly k.

Let $\mathbf{CF}^1_{cb}$ and $\mathbf{GS}^1_{cb}$ be the language families generated by context-free and arbitrary grammars with capacity function **1**. Then,

Lemma 12. $\mathbf{CF}_{cb} = \mathbf{CF}^1_{cb}$ *and* $\mathbf{GS}_{cb} = \mathbf{GS}^1_{cb}$.

On the other hand, capacity-bounded GS grammars generate exactly the family of matrix languages of finite index. This is in contrast to derivation bounded grammars which generate only context-free languages of finite index [32].

Lemma 13. $\mathbf{GS}_{cb} = \mathbf{MAT}_{fin}$.

It turns out that capacity-bounded context-free grammars are strictly between context-free languages of finite index and matrix languages of finite index.

Theorem 14. $\mathbf{CF}_{fin} \subset \mathbf{CF}_{cb} \subset \mathbf{GS}_{cb} = \mathbf{MAT}_{fin}$.

The next theorem shows that the families of capacity bounded matrix and vector languages are exactly the family of matrix languages with finite index.

Theorem 15. $\mathbf{MAT}_{fin} = \mathbf{VEC}^{[\lambda]}_{cb} = \mathbf{MAT}^{[\lambda]}_{cb}$.

As regards closure properties, we remark that the constructions showing the closure of **CF** under homomorphisms, union, concatenation and Kleene closure can be easily extended to the case of capacity-bounded languages.

Theorem 16. $\mathbf{CF}_{cb}$ *is closed under homomorphisms, union, concatenation and Kleene closure.*

Regarding intersection with regular sets and inverse homomorphisms, we can show non-closure properties.

Theorem 17. $\mathbf{CF}_{cb}$ *is neither closed under intersection with regular sets nor under inverse homomorphisms.*

Control by Petri nets can in a natural way be adapted to Petri nets with place capacities. A context-free grammar is controlled by its context-free Petri net with place capacity by only allowing derivations that correspond to valid firing sequences respecting the capacity bounds. The (trivial) proof for the equivalence between context-free grammars and grammars controlled by cf Petri nets can be immediately transferred to context-free grammars and Petri nets with capacities:

Theorem 18. *Grammars controlled by context-free Petri nets with place capacity functions generate the family of capacity-bounded context-free languages.*

Let us now turn to grammars controlled by arbitrary Petri nets with capacities. Let $G = (V, \Sigma, S, R, N, \gamma, M)$ be an arbitrary Petri net controlled grammar. G is called a grammar controlled by an arbitrary Petri net with place capacity if N is a Petri net with place capacity. The families of languages generated by grammars controlled by arbitrary Petri nets with place capacities (with erasing rules) is denoted by $\mathbf{PN}_{cb}(x, y)$ ($\mathbf{PN}^{\lambda}_{cb}(x, y)$) where $x \in \{f, -\lambda, \lambda\}$ and $y \in \{r, t, g\}$.

The next statement indicates that the language generated by a grammar controlled by an arbitrary Petri net with place capacities iff it is generated by a matrix grammar (for details, see [56]).

Theorem 19. *For $x \in \{f, -\lambda, \lambda\}$ and $y \in \{r, t, g\}$,*

$$\mathbf{PN}_{cb}(x, y) = \mathbf{MAT} \subseteq \mathbf{PN}^{\lambda}_{cb}(x, y) = \mathbf{MAT}^{\lambda}.$$

We summarize our results in the following theorem.

Theorem 20. *The following inclusions and equalities hold:*

$$\begin{aligned}\mathbf{CF}_{fin} &\subset \mathbf{CF}_{cb} = \mathbf{CF}^{1}_{cb} \\ &\subset \mathbf{MAT}_{fin} = \mathbf{MAT}^{[\lambda]}_{cb} = \mathbf{VEC}^{[\lambda]}_{cb} = \mathbf{GS}_{cb} = \mathbf{GS}^{1}_{cb} \\ &\subset \mathbf{MAT} = \mathbf{PN}_{cb}(x, y) \subseteq \mathbf{MAT}^{\lambda} = \mathbf{PN}^{\lambda}_{cb}(x, y)\end{aligned}$$

where $x \in \{f, -\lambda, \lambda\}$ and $y \in \{r, t, g\}$.

7. Conclusions and future research

The chapter summarizes the recent results on Petri net controlled grammars presented in [18–23, 56, 59–62] and the close related topic: capacity-bounded grammars. Though the theme of regulated grammars is one of the classic topics in formal language theory, a Petri net controlled grammar is still interesting subject for the investigation for many reasons. On the one hand, this type of grammars can successfully be used in modeling new problems emerging in manufacturing systems, systems biology and other areas. On the other hand, the graphically illustrability, the ability to represent both a grammar and its control in one structure, and the possibility to unify different regulated rewritings make this formalization attractive for the study. Moreover, control by Petri nets introduces the concept of *concurrency* in regulated rewriting systems.

We should mention that there are some open problems, the study of which is of interest: one of them concerns to the classic open problem of the theory of regulated rewriting systems – the strictness of the inclusion $\mathbf{MAT} \subseteq \mathbf{MAT}^{\lambda}$. We showed that language families generated by (arbitrary) Petri net controlled grammars are between the families **MAT** and $\mathbf{MAT}^{\lambda}$. Moreover, the work [65] of G. Zetzsche shows that the erasing rules in Petri net controlled grammars with finite set of final markings can be eliminated without effecting on the generative power, which gives hope that one can solve this problem.

There is also another very interesting topic in this direction for the future study. If we notice the definitions of derivation-bounded [32] or nonterminal-bounded grammars [3–5] only nonterminal strings are allowed as left-hand sides of production rules. Here, an interesting

question is emerged, what kind of languages can be generated if we derestrict this condition, i.e., allow any string in the left-hand side of the rules?

In all investigated types of Petri net controlled grammars, we only used the sequential firing mode of transitions. The consideration of simultaneous firing of transitions, another fundamental feature of Petri nets, opens a new direction for the future research: one can study *grammars controlled by Petri nets under parallel firing strategy*, which introduces concurrently parallelism in formal language theory.

Grammar systems can be considered as a formal model for a phenomenon of solving a given problem by dividing it into subproblems (grammars) to be solved by several parts in turn (CD grammar systems) or in parallel (PC grammar systems). The control of derivations in grammar systems also allows increasing computational power grammar systems. We can extend the regulation of a rule by a transition to the regulation a set of rules by a transition, which defines a new type of grammar systems: the firing of a transition allows applying several (assigned) rules in a derivation step parallelly and different modes.

In [19–22, 41, 62] it was shown that by adding places and arcs which satisfy some structural requirements one can generate well-known families of languages as random context languages, valence languages, vector languages and matrix languages. Thus, the control by Petri nets can be considered as a unifying approach to different types of control. On the other hand, Petri nets can be transformed into *occurrence nets*, i.e., usually an infinite, tree-like structure whose nodes have the same labels as those of the places and transitions of the Petri net preserving the relationship of adjacency, using *unfolding technique* introduced in [43] and given in [24] in detail under the name of *branching processes*. Any *finite initial* part, i.e., *prefix* of the occurrence net of a cf Petri net can be considered as a derivation tree for the corresponding context-free grammar as it has the same structure as a usual derivation tree, here we can also accept the rule of reading "leaf"-places with tokens from the left to the right as in usual derivation trees. We can also generalize this idea for regulated grammars considering prefixes of the occurrences nets obtained from cf Petri nets with additional places. Hence, we can take into consideration the grammar as well as its control, and construct (Petri net) derivation trees for regulated grammars, which help to construct effective parsing algorithms for regulated rewriting systems. Though the preliminary results (general parsing algorithms, Early-like parsing algorithm for deterministic extended context-free Petri net controlled grammars, etc.) were obtained in [63, 64], the problem of the development of the effective parsing algorithms for regulated grammars remain open.

Acknowledgements

This work has been supported by Ministry of Higher Education of Malaysia via Fundamental Research Grant Scheme FRGS /1/11/SG/UPM/01/1 and Universiti Putra Malaysia via RUGS 05-01-10-0896RU/F1.

Author details

Jürgen Dassow and Ralf Stiebe
Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, Magdeburg, Germany

Gairatzhan Mavlankulov, Mohamed Othman, Mohd Hasan Selamat and Sherzod Turaev
Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, UPM Serdang, Selangor, Malaysia

8. References

[1] Abraham, A. [1965]. Some questions of phrase-structure grammars, *Comput. Linguistics* 4: 61–70.
[2] Al-A'ali, M., Khan, A. & Al-Shamlan, N. [1996]. Simplification of context-free grammar through Petri net, *Computers and Structures* 58: 1055–1058.
[3] Altman, E. [1964]. The concept of finite representability, *Systems Research Center Report SRC 56-A-64-20*, Case Institute of Technology.
[4] Altman, E. & Banerji, R. [1965]. Some problems of finite representability, *Information and Control* 8: 251–263.
[5] Banerji, R. [1963]. Phrase structure languages, finite machines, and channel capacity, *Information and Control* 6: 153–162.
[6] Beek, M. t. & Kleijn, H. [2002]. Petri net control for grammar systems, *Formal and Natural Computing*, Vol. 2300 of *LNCS*, Springer, pp. 220–243.
[7] Brainerd, B. [1968]. An analog of a theorem about context-free languages, *Information and Control* 11: 561–567.
[8] Chomsky, N. [1956]. Three models for the description of languages, *IRE Trans. on Information Theory* 2(3): 113–124.
[9] Chomsky, N. [1957]. *Syntactic structure*, Mouton, Gravenhage.
[10] Chomsky, N. [1959]. On certain formal properties of grammars, *Information and Control* 2: 137–167.
[11] Crespi-Reghizzi, S. & Mandrioli, D. [1974]. Petri nets and commutative grammars, *Internal Report 74-5*, Laboraterio di Calcolatori, Instituto di Elettrotecnica ed Elettromca del Politecnico di Milano, Italy.
[12] Crespi-Reghizzi, S. & Mandrioli, D. [1975]. Properties of firing sequences, *Proc. MIT Conf. Petri Nets and Related Methods*, MIT, Cambridge, Mass., pp. 233–240.
[13] Crespi-Reghizzi, S. & Mandrioli, D. [1977]. Petri nets and Szilard languages, *Inform. and Control* 33: 177–192.
[14] Darondeau, P. [2001]. On the Petri net realization of context-free graphs, *Theor. Computer Sci.* 258: 573–598.
[15] Dassow, J. [1988]. Subregularly controlled derivations: Context-free case, *Rostock. Math. Kolloq.* 34: 61–70.
[16] Dassow, J. & Păun, G. [1989]. *Regulated rewriting in formal language theory*, Springer-Verlag, Berlin.
[17] Dassow, J. & Truthe, B. [2008]. Subregularly tree controlled grammars and languages, *in* E. Csuhaj-Varjú & Z. Esik (eds), *Proc. the 12th International Conference AFL 2008*, Balatonfured, Hungary, pp. 158–169.
[18] Dassow, J. & Turaev, S. [2008a]. Arbitrary Petri net controlled grammars, *in* G. Bel-Enguix & M. Jiménez-López (eds), *Linguistics and Formal Languages. Second International Workshop on Non-Classical Formal Languages In Linguistics, Tarragona, Spain*, pp. 27–39. ISBN 978-84-612-6451-3.
[19] Dassow, J. & Turaev, S. [2008b]. *k*-Petri net controlled grammars, *in* C. Martín-Vide, F. Otto & H. Fernau (eds), *Language and Automata Theory and Applications. Second*

International Conference, LATA 2008. Revised Papers, Vol. 5196 of *LNCS*, Springer, pp. 209–220.

[20] Dassow, J. & Turaev, S. [2009a]. Grammars controlled by special Petri nets, *in* A. Dediu, A.-M. Ionescu & C. Martín-Vide (eds), *Language and Automata Theory and Applications, Third International Conference, LATA 2009*, Vol. 5457 of *LNCS*, Springer, pp. 326–337.

[21] Dassow, J. & Turaev, S. [2009b]. Petri net controlled grammars: the case of special Petri nets, *Journal of Universal Computer Science* 15(14): 2808–2835.

[22] Dassow, J. & Turaev, S. [2009c]. Petri net controlled grammars: the power of labeling and final markings, *Romanian Jour. of Information Science and Technology* 12(2): 191–207.

[23] Dassow, J. & Turaev, S. [2010]. Petri net controlled grammars with a bounded number of additional places, *Acta Cybernetica* 19: 609–634.

[24] Engelfriet, J. [1991]. Branching processes of petri nets, *Acta Informatica* 28: 575–591.

[25] Erqing, X. [2004]. A Pr/T-Net model for context-free language parsing, *Fifth World Congress on Intelligent Control and Automation*, Vol. 3, pp. 1919–1922.

[26] Farwer, B., Jantzen, M., Kudlek, M., Rölke, H. & Zetzsche, G. [2008]. Petri net controlled finite automata, *Fundamenta Informaticae* 85(1-4): 111–121.

[27] Farwer, B., Kudlek, M. & Rölke, H. [2006]. Petri-net-controlled machine models, *Technical Report 274*, FBI-Bericht, Hamburg.

[28] Farwer, B., Kudlek, M. & Rölke, H. [2007]. Concurrent Turing machines, *Fundamenta Informaticae* 79(3-4): 303–317.

[29] Fernau, H. & Holzer, M. [1997]. Conditional context-free languages of finite index, *New Trends in Formal Languages – Control, Cooperation, and Combinatorics (to Jürgen Dassow on the occasion of his 50th birthday)*, Vol. 1218 of *LNCS*, pp. 10–26.

[30] Fernau, H. & Holzer, M. [2008]. Regulated finite index language families collapse, *Technical Report WSI-96-16*, Universität Türbingen, Wilhelm-Schickard-Institut für Informatik.

[31] Ginsburg, S. & Spanier, E. [1968a]. Contol sets on grammars, *Math. Syst. Th.* 2: 159–177.

[32] Ginsburg, S. & Spanier, E. [1968b]. Derivation bounded languages, *J. Comput. Syst. Sci.* 2: 228–250.

[33] Hack, M. [1975a]. *Decidablity questions for Petri nets*, PhD thesis, Dept. of Electrical Engineering, MIT.

[34] Hack, M. [1975b]. Petri net languages, *Computation Structures Group Memo, Project MAC 124*, MIT, Cambridge Mass.

[35] Hauschildt, D. & Jantzen, M. [1994]. Petri nets algorithms in the theory of matrix grammars, *Acta Informatica* 31: 719–728.

[36] Hopcroft, J. & Ullman, J. [1990]. *Introduction to automata theory, languages, and computation*, Addison-Wesley Longman Publishing Co., Inc.

[37] Ibarra, O. [1970]. Simple matrix grammars, *Inform. Control* 17: 359–394.

[38] Jantzen, M., Kudlek, M. & Zetzsche, G. [2008]. Language classes defined by concurrent finite automata, *Fundamenta Informaticae* 85(1-4): 267–280.

[39] Jiang, C. [1996]. Vector grammar and PN machine, *Sci. Chin. (Ser. A)* 24: 1315–1322.

[40] Liptop, R. [1976]. The reachability problem requires esponential space, *Technical Report 62*, Yale University.

[41] Marek, V. & Češka, M. [2001]. Petri nets and random-context grammars, *Proc. of the 35th Spring Conference: Modelling and Simulation of Systems*, MARQ Ostrava, Hardec nad Moravicí, pp. 145–152.

[42] Martín-Vide, C., Mitrana, V. & Păun, G. (eds) [2004]. *Formal languages and applications*, Springer-Verlag, Berlin.

[43] McMillan, K. [1995]. A technique of a state space search based on unfolding, *Formal Methods in System Design* 6(1): 45–65.
[44] Moriya, E. [1973]. Associate languages and derivational complexity of formal grammars and languages, *Information and Control* 22: 139–162.
[45] Murata, T. [1989]. Petri nets: Properties, analysis and applications, *Proceedings of the IEEE* 77(4): 541–580.
[46] Peterson, J. [1976]. Computation sequence sets, *J. Computer and System Sciences* 13: 1–24.
[47] Peterson, J. [1981]. *Petri net theory and modeling of systems*, Prentice-Hall, Englewood Cliffs, NJ.
[48] Păun, G. [1977]. On the index of grammars and languages, *Inf. Contr.* 35: 259–266.
[49] Păun, G. [1979]. On the family of finite index matrix languages, *JCSS* 18(3): 267–280.
[50] Reisig, W. & Rozenberg, G. (eds) [1998]. *Lectures on Petri Nets I: Basic Models*, Vol. 1491 of *LNCS*, Springer, Berlin.
[51] Rozenberg, G. [1976]. More on ET0L systems versus random context grammars, *IPL* 5(4): 102–106.
[52] Rozenberg, G. & Salomaa, A. (eds) [1997]. *Handbook of formal languages*, Vol. 1–3, Springer.
[53] Rozenberg, G. & Vermeir, D. [1978a]. On ET0L systems of finite index, *Inf. Contr.* 38: 103–133.
[54] Rozenberg, G. & Vermeir, D. [1978b]. On the effect of the finite index restriction on several families of grammars, *Inf. Contr.* 39: 284–302.
[55] Rozenberg, G. & Vermeir, D. [1978c]. On the effect of the finite index restriction on several families of grammars; Part 2: context dependent systems and grammars, *Foundations of Control Engineering* 3(3): 126–142.
[56] Selamat, M. & Turaev, S. [2010]. Grammars controlled by petri nets with place capacities, *2010 International Conference on Computer Research and Development*, pp. 51–55.
[57] Starke, P. [1978]. Free Petri net languages, *Mathematical Foundations of Computer Science 1978*, Vol. 64 of *LNCS*, Springer, Berlin, pp. 506–515.
[58] Starke, P. [1980]. *Petri-Netze*, Deutscher Verlag der Wissenschaften.
[59] Stiebe, R. & Turaev, S. [2009a]. Capacity bounded grammars, *Journal of Automata, Languages and Combinatorics* 15(1/2): 175–194.
[60] Stiebe, R. & Turaev, S. [2009b]. Capacity bounded grammars and Petri nets, *EPTCS* 3: 193–203.
[61] Stiebe, R. & Turaev, S. [2009c]. Capacity bounded grammars and Petri nets, *in* J. Dassow, G. Pighizzini & B. Truthe (eds), *Eleventh International Workshop on Descriptional Complexity of Formal Systems, Magdeburg, Germany*, pp. 247–258.
[62] Turaev, S. [2007]. Petri net controlled grammars, *Third Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, MEMICS 2007*, Znojmo, Czechia, pp. 233–240. ISBN 978-80-7355-077-6.
[63] Turaev, S., Krassovitskiy, A., Othman, M. & Selamat, M. [2011]. Parsing algorithms for grammars with regulated rewriting, *in* A. Zaharim, K. Sopian, N. Mostorakis & V. Mladenov (eds), *Recent Researches in Applied Informatics and Remote Sensing. The 11th WSEAS International Conference on APPLIED COMPUTER SCIENCE*, pp. 103–109.
[64] Turaev, S., Krassovitskiy, A., Othman, M. & Selamat, M. [2012]. Parsing algorithms for regulated grammars, *Mathematical Models & Methods in Applied Science* . (to appear).
[65] Zetzsche, G. [2009]. Erasing in petri net languages and matrix grammars, *Proceedings of the 13th International Conference on Developments in Language Theory*, DLT '09, Springer-Verlag, Berlin, Heidelberg, pp. 490–501.